NAGW-1333

REAL-TIME PLANNING AND INTELLIGENT CONTROL

By:

A.C. Sanderson
L.S. Homem de Mello

Department of Electrical, Computer and Systems Engineering
Department of Mechanical Engineering, Aeronautical Engineering & Mechanics
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

October 1988

CIRSSE Document #31

REAL–TIME PLANNING AND INTELLIGENT CONTROL

Arthur C. Sanderson

Electrical, Computer, and Systems Engineering Department,
Rensselaer Polytechnic Institute, Troy, NY 12180

Luiz S. Homem de Mello

Jet Propulsion Laboratory/California Institute of
Technology, Pasadena, CA 91109

NATO Advanced Research Workshop on Knowledge Based
Robot Control
October 10-12, 1988.

REAL-TIME PLANNING AND INTELLIGENT CONTROL

Arthur C. Sanderson

Electrical, Computer and Systems Engineering Department,
Rensselaer Polytechnic Institute, Troy, NY 12180
and

Luiz S. Homem de Mello

Jet Propulsion Laboratory/California Institute of Technology, Pasadena, CA 91109

ABSTRACT

Classical control systems utilize *implicit* analytical models and select inputs using numerical computation in order to achieve desired plant performance in real time. Artificial intelligence planning systems utilize *explicit* symbolic or logical models of discrete operations and construct consistent planned sequences using symbolic reasoning to achieve specified task goals. In the development of more complex intelligent systems, these two approaches must interact in order to successfully execute real-time operations in a manner consistent with the global goals and constraints of the task. Intelligent control and real-time planning are approaches to developing a theory and methodology for the design systems which incorporate this interface. In this paper, we provide some examples of the role of task representation in the definition of these control and planning strategies. We describe in detail some results on task planning of assembly sequences and the use of precedence relations as an implicit model of ordering constraints for assembly operations.

1. INTRODUCTION

The development of increasingly complex machines and systems has focused attention on principles of representation, planning, and control, and their implementation into architectures which will support intelligent decision making and task execution behavior. The complexity of such a system increases as the task transcends a single domain of sensing or action, and it becomes necessary to construct representations combining elementary sensing and actions into multiple levels of abstraction. Within this framework, a robot motion task such as "move point *a* to point *b*" is a "simple" task in the sense that the decisions required to initiate and execute the robot motion remain within the domain of the motion state space parameters. While the required control algorithm necessary to execute the motion subject to disturbances may be sophisticated, the formulation of the problem and the control algorithm remain within a single representational domain.

Representation of the robot task "put the *book* on the *table*" transcends the motion space domain of the robot. This task specification introduces objects, relations, constraints, and

assumptions which cannot be directly represented in the motion space of the robot. In this case, a set of abstractions, usually represented by symbols, are introduced into the representation in order to adequately describe the task and its relationship to the physical and geometric state of the system. In this case, both sensing and actions must be expressed in terms of this abstract space, and the planning of sequences of correct and consistent actions becomes the focus of the decision. The control of actions in any single motion domain then becomes a subset of the planned sequence of actions at the more abstract level. This interface between the planning and control functions in a problem domain with hierarchical representation is fundamental to the development of new generations of intelligent systems.

Intelligent systems make decisions about the actions which they undertake in order to reach stated goals [1]. Such decisions can be made most effectively when a prediction of the consequences of actions is available. The strength of classical control theory has been the development of a methodology which uses analytical models of the dynamic behavior of a plant in order to predict the consequences of control actions and therefore choose the optimum inputs at any given moment. While such an analytical model may be used to explicitly predict the outcome of actions, that is, to generate a predicted trajectory for the plant, in classical control systems this analytical model is used implicitly rather than explicitly. The analytical model of the plant is used as a basis to derive a controller algorithm which directly computes new input actions which guarantee the outcome and performance of the control system. The controller algorithm utilizes the plant model implicitly and executes the numerical computations on-line in order to produce the correct inputs. Such an approach has tremendous advantages for real time implementation since it guarantees optimality of the next step input without exhaustive prediction of the trajectories.

Planning sequences of actions to accomplish predefined goals has been a topic of key interest in the development of artificial intelligence problem solving techniques [2,3,4,5,6]. In this literature, an action is represented by a symbol with specific preconditions defined for the execution of the action and specific consequences of the action. In most of the AI literature on domain independent planning, the state of the world has been defined by propositional logic, and the pre- and post-conditions for actions have been specified in terms of the same logical expressions. The key thrust of this work has been the formulation of the planning problem as a search problem over the space of feasible action sequences. Much of this planning work has recognized that real planning problems generate enormous search spaces and that techniques such as subgoal generation, hierarchies, and nonlinear, or partially ordered, plans may be necessary to make this search problem traceable. In practice many of these planning techniques require the incorporation of heuristics in order to obtain solutions.

AI planning techniques are thought of as "intelligent" because they operate on an abstract or global representation of the world. Based on such a global representation it should be possible to reason about new or unpredicted situations in order to derive plans which were not anticipated at the time of the formulation of the problem. While the availability of such a global knowledge base does not guarantee intelligent behavior of the system, it suggests that an appropriate planner with access to representation of the world at this level of abstraction should be able to generate well developed plans for new situations. The heuristic search approach to AI planning generates intelligent plans, that is, they incorporate global knowledge. This approach also generates explicit plans. The resulting plan is an *explicit* sequence of actions which has been tested in the search process for consistency and optimality based on a definition

of the current state and the goal state. Any change in the assumptions or constraints about the problem will require a new search, sequence generation, and verification. Such explicit plan generation is not conducive to real-time operation.

2. REAL-TIME PLANNING AND INTELLIGENT CONTROL

This incompatibility between the implicit, real-time, numerical control algorithm and the explicit, off-line, symbolic planning algorithm has led to a fundamental dilemma in the development of architectures for complex dynamic systems. This dilemma is illustrated in Figure 1, which characterizes approaches to planning and control in complex systems. Two different axes describe these approaches. The representation space of actions may be either specific to an elementary action problem domain (local), such as robot motion, or may be generalized to represent the abstract sets of actions in a global problem domain, such as in robot object relationships and interactions. The representation of local or elementary actions typically maps into a *numeric* description of a single well-defined state space. The global or abstract actions typically map into *symbolic* or logical representations. An intelligent system utilizes a predictive model of the world as a basis for decision-making about the sequence of actions which it executes. This predictive model may be incorporated as an *implicit* part of the decision-making algorithm, in which case the next action is chosen without explicit generation of the full sequence of actions to the goal, or the predictive model may be *explicit* in the decision-making algorithm, in which case the full sequence of actions is generated each time a new plan is required.

The table shown in Figure 1 therefore defines a space for the categorization of approaches to planning and control in complex systems. Classical, model-based control utilizes a *numeric* representation of elementary actions in a well-defined state space along with an *implicit* model of the consequences of those actions in order to provide an efficient decision-making algorithm which guarantees certain performance characteristics of the resulting system. On the other hand, an AI planning system utilizes *symbolic* representation of global or abstract actions and *explicit* search over possible sequences of actions as a decision-making tool in the generation of plans.

The other two blocks in Figure 1 offer insight into the nature of the interaction between planning and control concepts. The *symbolic/implicit* block defines a set of algorithms which have been approached in two different ways. Work on intelligent control concepts is focused on the extension of classical model based control to incorporate symbolic reasoning systems, in order to make real time decisions among alternative control policies. Such a layered intelligent control system has most often retained the implicit analytical model for elementary actions and layered an explicit symbolic model, such as a rule-based system, which selects, in real-time, alternative controller policies based on current state information. The implicit model is most often incorporated into a heuristic rule set which avoids the explicit generation of plans, and the symbolic reasoning often choses among alternative control strategies rather than discrete actions hoping to exploit the optimality offered by the implicit model in the local domain.

Work on real-time planning has attempted to extend the AI approach to incorporate

	NUMERIC (Local or Elementary Action Problem Domain)	SYMBOLIC (Global or Abstract Action Domain)
IMPLICIT MODEL	Model-Based Control	Intelligent Control/ Real-time Planning
EXPLICIT MODEL	Dynamic Programming/ Simulation-based Control	AI Planning

Figure 1 Approaches to planning and control in complex systems.

reactive properties based on observation or sensing of current state information. Such a real-time planning capability requires projection or prediction within the planner and monitoring and matching of current state information relative to the predicted plan. In the simplest case, discrepancy between the predicted plan and the observed system state requires complete replanning from the new current state to the goal. More efficient approaches recognize classes of disturbances (which might be specific to the problem domain) and permit reasoning about alterations to the plan rather than complete sequence regeneration. Such a reactive planner often works within the scope of a fixed projection length in the plan generation process. More general plans and incomplete, skeletal, or partially ordered plans may be generated initially and provide the global framework for implicit decisions about the execution of specific action sequences.

The *explicit/numeric* block in Figure 1 is also of considerable interest. In conventional terms one may think of a complex plant for which no adequate analytical model is available. If a numerical simulation of the plant can be generated, then a controller over the elementary actions or inputs to the plant might be developed based on the utilization of the simulation tool as an explicit model of the consequences of input actions. Such an approach clearly raises difficulties from a classical control standpoint since it is difficult to guarantee the stability or other performance properties of the resulting system. A more common version of this problem may be found in the classical dynamic programming approach to control. In dynamic programming, given an explicit model of the state space, the control problem is set up, the performance measures attached to each of the states, and control actions link the states. The dynamic programming algorithm then provides an efficient means to search for an optimal path among the alternative sequences of control actions. The properties of the numerical search space permit the implementation of an efficient search algorithm which minimizes back tracking and permits local decisions once the performance map over the state

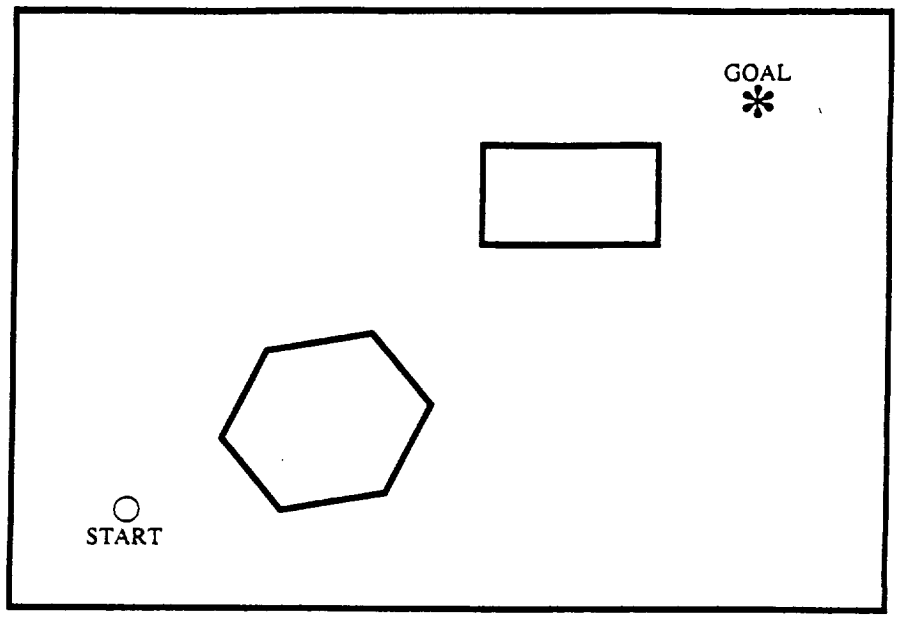


Figure 2 Robot path planning task.

space has been generated. In addition, there exists a continuum of approaches which permit this performance state space to be completely or partially generated using analytical model or approximations and, therefore, combine the dynamic programming approach to search with model based control.

The development of complex intelligent systems will require the linking together of many of the techniques represented in Figure 1. In robot systems these issues arise at several different levels. Kinematic and dynamic models form the basis for most robot motion control algorithms. Their implicit models are available for the generation of the controller. When the robot is integrated into more complex environments, the complexity of the disturbances, the interpretation of sensing information, and the physical characteristics of the arm itself may make it more difficult to analyze and develop pure model based control. An obstacle avoidance task, such as that shown in Figure 2, is one example of this more complicated environment. The task of the robot is to move from point *START* to point *GOAL* without colliding with the obstacles. The local kinematic and dynamic characteristics of the robot are known, and a controller for local path motion has been defined. Several different approaches have been suggested in the literature to this type of problem. These distinct approaches illustrate a number of the issues described above.

One strategy for this robot path planning problem might use a purely geometric search. Random search on a grid of points will guarantee a solution and, given enough time, will guarantee that an exhaustive search would yield an optimal solution. However, an analysis of this search domain yields some insight into the nature of the paths which occur between

the initial point and the goal point. Normally, every shortest path trajectory in this domain must past through either the initial point, the goal point, or one or more of the vertices of the polygon shaped obstacles. This observation has been used extensively in the literature on path planning among obstacles [7]. A solution such as shown in Figure 3, may be found by a search over the much more modest space of alternatives exhibited by this so called visibility graph. The resulting path is an explicit plan generated by a discrete search over the space of alternatives. Any small change in the positions of the obstacles or the initial or goal points would require complete replanning of the path in order to guarantee an optimal solution.

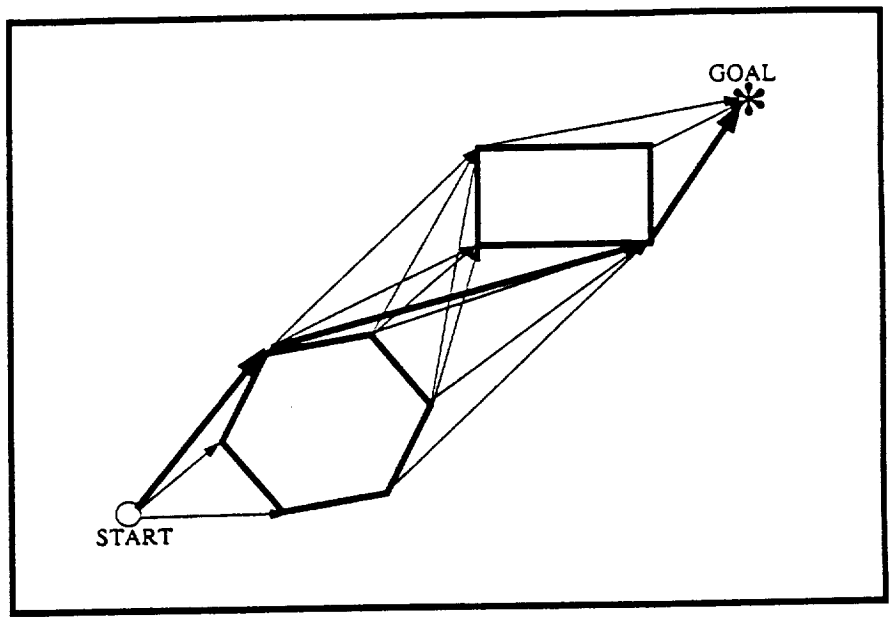


Figure 3 Discrete search over feasible paths using the visibility graph.

A different representation of this problem space suggests an alternative approach to solution. By considering the problem space as a continuous space in which obstacles are regions to be avoided, one may develop potential function mapping algorithms which relate the geometry of the configuration of the obstacles to a continuous function over the state space. Such an approach has been generalized by Khatib [8] to consider motion and force control of manipulators. A potential function representation shown in Figure 4 lends itself to the generation of implicit rather than explicit control algorithms. An implicit algorithm which seeks an energy minimum at the goal point, yet avoids the energy maxima formed by the obstacles, may be easily developed. Such a control algorithm decides on successive local motions based only on the local model of the potential field. While the resulting path does not guarantee the shortest path, the resulting paths may be adequate or even advantageous in a dynamic or uncertain environment. In addition, changes in the environment may be

reflected in perturbations of the potential field and, therefore, require only local rather than global recomputation in the field and no change at all in the implicit control algorithm.

An interesting combination of these algorithms is shown in Figure 5. In this approach, a distance metric space field [9] is defined instead of the potential field shown previously. In this approach, every point in the space is mapped into a continuous metric which corresponds to the shortest distance between that point and the goal point. While this field itself is sensitive to the positions of obstacles, the ability to compute such a field, using local parallel operators, has significant advantages. The resulting controller algorithm is based only on local model information, that is, a controller which follows the gradient of the distance metric space will determine the shortest path between the initial point and the goal point. The resulting optimal trajectory is therefore the same as that derived from the global path planner. This distant metric space approach may be combined with the visibility graph to provide a real time implicit controller which also has global knowledge of the line segment sequence which constitutes the optimal path.

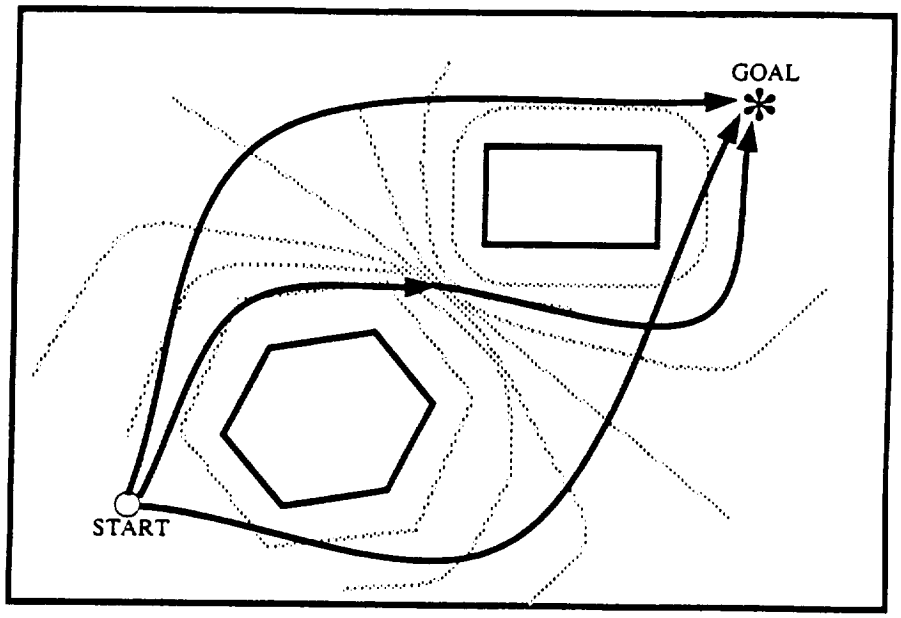


Figure 4 Potential field repels from obstacles and attracts to goal.

These examples of motion planning and path planning indicate some of the trade-offs and alternative strategies which may be employed in approaching the same problem. Task planning incorporates additional complexity into the problem because the specification of the goal state itself is usually at a level of abstraction beyond the motion or kinematic description of the robot. In such cases, the representation of the problem domain must incorporate the decomposition of the task into elementary perceptions and actions. The planner then

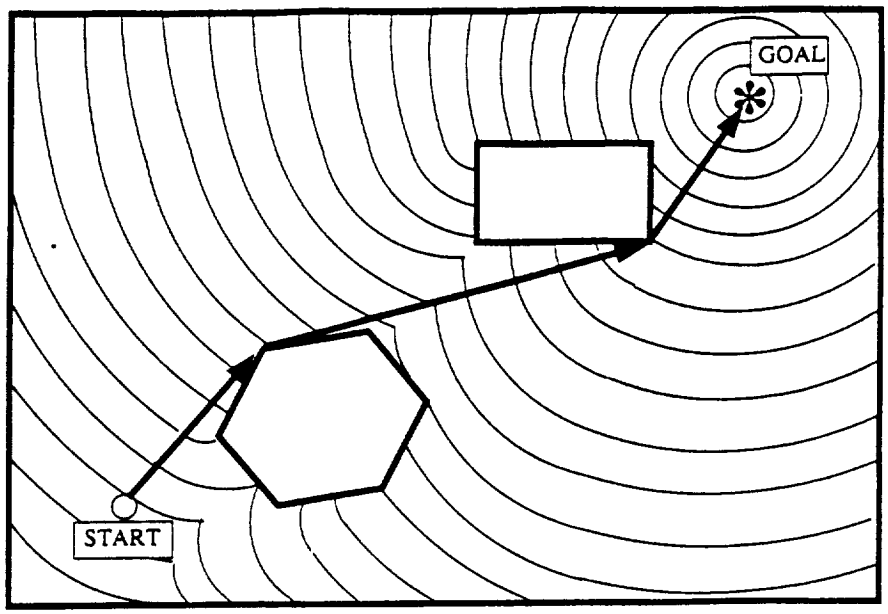


Figure 5 Distance metric space representation permits tracking of minimum distance path using implicit algorithm for local gradient.

must sequence these actions at the abstract level and coordinate the control and execution of perceptions and actions at all levels. Such planned sequences of actions are most often represented explicitly by the sequences themselves. The planning process generates a search tree over the feasible states of the system, and a search algorithm operates on that state tree in order to select desired sequences. The decomposition of the problem in the determination of the nature and size of this search tree is a critical factor in the efficiency of the planning process. In addition, it may be possible to generate more compact, implicit, representations of feasible action sequences using either methods of finite automata models, grammars, or precedence relations. Finite automata typically capture the state transition characteristics which can describe the feasible action sequences. Equivalently such automata models can be represented as formal grammars. Problems which are restricted to finite length sequences often do not lend themselves to simple grammatical representations, and general precedence relations may be more appropriate. In the next section we summarize some results on the generation of precedence relations as implicit representations of action sequences for assembly tasks.

3. PRECEDENCE RELATIONS FOR ASSEMBLY SEQUENCE PLANNING

In assembly task planning [10–15], a sequence of operations for parts and subassemblies

is generated. The most desirable assembly sequence may depend upon a variety of factors, including the available resources, the time cost of certain operations, or the reliability of the operation itself. While alternative plans must be generated and evaluated in an off-line mode for many manufacturing tasks, it is also common that real-time modification or replanning must occur in order to effectively utilize manufacturing resources or to carry out operations such as maintenance or repair of products. An *implicit* representation of plans is therefore desirable for these types of applications.

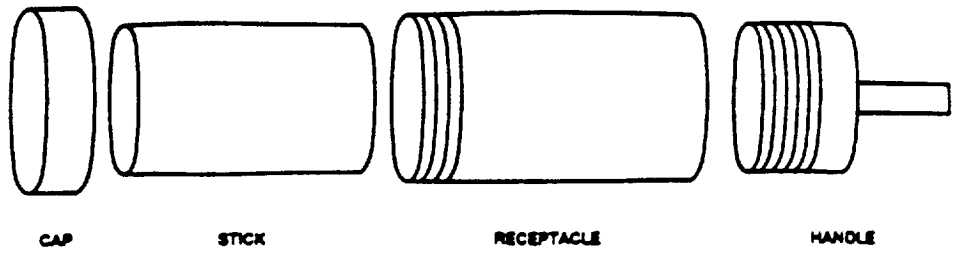
An assembly sequence plan is a fixed length sequence, and may be represented either by the sequence of assembly state partitions or by the sequence of parts connections. In our work on assembly sequence planning [10–15], we have introduced the AND/OR graph representation of assembly plans as an efficient representation of the feasible operation sequences available to accomplish the assembly goal. We have shown that this AND/OR graph representation is complete and correct based on the definition of feasibility tests used in the generation algorithm and that this AND/OR graph is equivalent to the tree of assembly states. We can utilize the AND/OR graph as a basis for the search over alternative feasible assembly plans in order to chose the most desirable plan for a given application. However, the AND/OR graph is an *explicit* representation of plans in the sense that replanning, in general, requires a regeneration of the whole search process.

An alternative representation based on precedence relationships between operations may be introduced and provides an *implicit* relationship of the sequential properties which are characteristic of feasible assembly sequences for a given product [13,15,16,17]. While a set of such precedence relationships may always be found as described below, the simplicity of the representation depends strongly on the particular problem. An assembly with strong sequential constraints will often result in a set of precedence relationships which constitute a relatively simple set of implicit rules regarding the proper sequential ordering of operations. However, an assembly without such strong constraints may produce a large set of precedence relationships which become extremely cumbersome to evaluate on a real time basis.

For purposes of this discussion, we will consider an assembly to consist of a set of parts with relationships defined by a graph of connections as shown in Figure 6. Each connection, c_i, indicates a set of real or virtual contacts between two parts. The assembly process then consists of a series of operations which successfully form the connections shown in the graph of connections subject to constraints imposed by the geometric and mechanical feasibility of the operations together with the inherent constraints imposed by the simultaneous formation of connections due to the assembly graph itself. The AND/OR graph of assembly plans for this product is shown in Figure 7.

We will represent the state of the assembly process, s_i, by an L-dimensional binary vector $x = [x_1, x_2, ..., x_l]$ in which the ith component x_i is true or false, respectively if the ith connection is established in that state or not. We will use the notation $c_i < c_j$ to indicate the fact that the establishment of connection c_i must precede the establishment of connection c_j. We will use the notation $c_i \leq c_j$ to indicate the fact that the establishment of connection c_i must precede or be simultaneous with the establishment of connection c_j. Furthermore, we will use a compact logical notation for logical combinations of precedence relationships; for example, we will write $c_i < c_j \cdot c_k$ when we mean $(c_i < c_j) \wedge (c_i < c_k)$, and we will write $c_i + c_j < c_k$ when we mean $(c_i < c_k) \vee (c_j < c_k)$.

An assembly sequence whose representation as an ordered sequence of binary vectors is $(\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N)$ and whose representation as an ordered sequence of subsets of connections is $(\gamma_1, \gamma_2, \ldots, \gamma_{N-1})$ satisfies the precedence relationship $c_i < c_j$ if $c_i \in \gamma_a, c_j \in \gamma_b$, and $a < b$. Similarly, the sequence satisfies $c_i \leq c_j$ if $c_i \in \gamma_a, c_j \in \gamma_b$, and $a \leq b$. For example, for the assembly shown in figure 6, the assembly sequence representation as an ordered sequence of binary vectors is



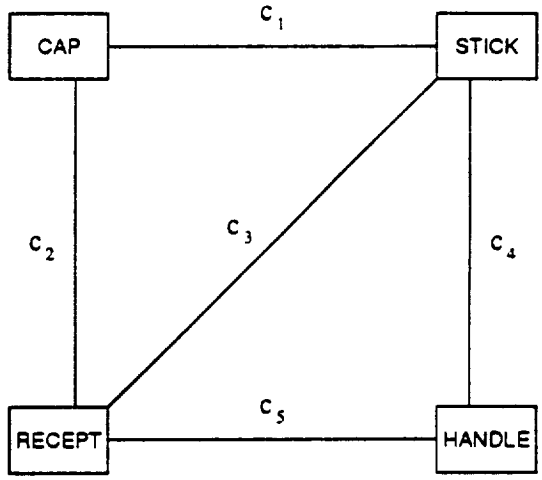


Figure 6 Graph of connections for simple assembly product.

([false, false, false, false, false]
[true, false, false, false, false]
[true, true, true, false, false]
[true, true, true, true, true]).

The corresponding representation as an ordered sequence of subsets of connections is *({c1} {c2, c3} {c4, c5})* satisfies the precedence relationships $c_2 < c_4$ and $c_2 \leq c_3$ but does not satsify the precedence relationships $c_2 < c_3$ and $c_2 \leq c_1$.

Each feasible assembly sequence of a given assembly can be uniquely characterized by a logical expression consisting of the conjunction of precedence relationships between the establishment of one connection and the establishment of another connection. For example, for the assembly shown in figure 6, the assembly sequence described in the previous paragraph can be uniquely characterized by the following conjunction of precedence relationships:$(c_1 < c_2) \wedge (c_2 < c_3) \wedge (c_3 < c_4) \wedge (c_4 < c_5)$.

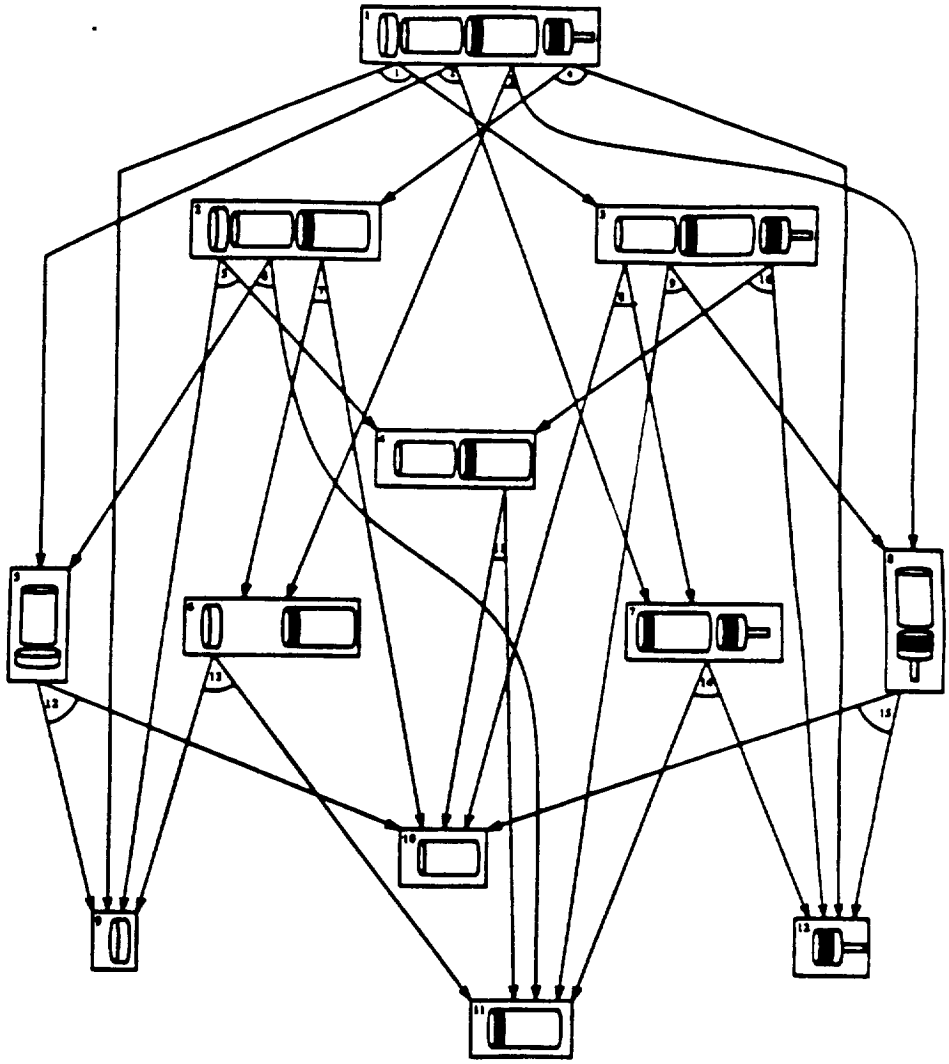


Figure 7 AND/OR graph representation of feasible assembly plans for product in figure 6.

The set of M feasible assembly sequences can be uniquely characterized by a disjunction of M conjunctions of precedence relationships in which each conjunction characterizes one assembly sequence. Clearly, this logical combination of precedence relationships constituttes a correct and complete representation for the set of all assembly sequences.

It is often possible to simplify this logical combination of precedence relationships using the rules of boolean algebra. Further simplification is possible if one notices that

there are logical combinations of precedence relationships that cannopt be satisfid by any assembly sequence. For the assembly shown in figure 1, for example, the combination $(c_1 < c_2) \wedge (c_2 < c_4) \wedge (c_2 \leq c_3) \wedge (c_4 \leq c_5) \wedge (c_5 \leq c_4)$ cannot be satisfied by any assembly sequence. These combinations can be set as don't care conditions in the simplification of the logical combination if precedence relationships.

Whenever the assembly has the two properties described below, it is possible to obtain a simpler precedence relationship of all assembly sequences. This representation will be obtained using the result of Theorem 1 below.

The first property is:

Property 1: Given any two states s_i and s_j , not necessasrily in the same assembly sequence, let γ_i and γ_j be the sets of connections that are established in assembly tasks τ_i and τ_j from s_i and s_j respectively. If

$\langle P, C_i \rangle$is the state's graph of connections associated to s_i ,

$\langle P, C_j \rangle$is the state's graph of connections associated to s_j,

$\gamma_i \subseteq \gamma_j$,

$C_i \subseteq C_j$, and

τ_j is geometrically and mechanically feasible,

then

τ_i is geometrically and mechanically feasible.

This property corresponds to the fact that if it is geometrically and mechanically feasible to establish a set of connections (γ_j) when many other connections (C_j) have already been established, then it is also geometrically and mechanically feasible to establish fewer connections ($\gamma_i \subseteq \gamma_j$) when fewer connections ($C_i \subseteq C_j$) have been established. Although many common assemblies have this property, there are assemblies that don't have it.

The second property is:

Property 2: If the subsets θ_1, θ_2,...,θ_k of the set of parts P characterize stable subassemblies, then these subsets are also stable within the assembly characterized by the set $\theta = \theta_1 \cup \theta_2 \cup \ldots \cup \theta_k$. That is, if all the subassemblies are stable, then the stability of the assembly is determined only by the stability of the connections between subassemblies, and not within subassemblies.

As in the case of Property 1, many common assemblies have this property, but there are simple examples which do not.

Theorem 1: Given an assembly made up of N parts whose graph of connections is $\langle P, C \rangle$ (with $C = \{c_1, c_2, \ldots, c_L\}$), let

$\{(\gamma_{11}\gamma_{21}\cdots\gamma_{(N-1)1}), (\gamma_{12}\gamma_{22}\cdots\gamma_{(N-1)2}), \ldots, (\gamma_{1M}\gamma_{2M}\cdots\gamma_{(N-1)M})\}$ be a set of M ordered sequences of subsets of connections that represent feasible assembly sequences. If the assembly has properties 1 and 2, then any ordered sequence of N-1 subsets of connections that represents an assembly sequence corresponds to a feasible assembly sequence if it satisfies the set of 2L precedence relationships:

$$c_i \leq \sum_{j=1}^{M} T_{ij} \quad i = 1, 2, \ldots, L \quad \text{and} \quad \sum_{j=1}^{M} H_{ij} \leq c_i \quad i = 1, 2, \ldots, L$$

where

$$T_{ij} = \prod_{k=1}^{L} \lambda_{ik} \quad with \quad \lambda_{ik} = \begin{cases} c_k \; if \; c_k \in \gamma_{lj} \; and \; l \geq i \\ true \; otherwise \end{cases}$$

$$H_{ij} = \prod_{k=1}^{L} \lambda_{ik} \quad with \quad \lambda_{ik} = \begin{cases} c_k \; if \; c_k \in \lambda_{lj} \; and \; l \leq i \\ true \; otherwise. \end{cases}$$

The sum and the product in this theorem are the logical operations OR and AND respectively. Each term T_{ij} (for i=1,2,...,L and for j=1,2,...,M) is the product of the variables corresponding to the connections that are established at the same time or after the establishment of connection c_i in the jth sequence. Similarly, each term H_{ij} is the product of the variables corresponding to the connections that are established before the establishment of connection c_i in the jth sequence. Precedence relationships that have "true" on either side are always satisfied. The proof of this theorem is presented elsewhere [13].

An example of the use of theorem 1 is based on the assembly in figure 6. This assembly has both properties 1 and 2. For that assembly, the set of feasible sequences is:

$$\begin{array}{lll} (\{c_1\}\{c_2,c_3\}\{c_4,c_5\}) & (\{c_1\}\{c_5\}\{c_2,c_3,c_4\}) & (\{c_2\}\{c_1,c_3\}\{c_4,c_5\}) \\ (\{c_2\}\{c_4\}\{c_1,c_3,c_5\}) & (\{c_3\}\{c_1,c_2\}\{c_4,c_5\}) & (\{c_3\}\{c_4,c_5\}\{c_1,c_2\}) \\ (\{c_4\}\{c_2\}\{c_1,c_3,c_5\}) & (\{c_4\}\{c_3,c_5\}\{c_1,c_2\}) & (\{c_5\}\{c_1\}\{c_2,c_3,c_4\}) \\ & (\{c_5\}\{c_3,c_4\}\{c_1,c_2\}) & \end{array}$$

Applying the result of theorem 1 to the above set of feasible sequences, the precedence relationships having connection c_1 alone on one side are:

$$c_1 \leq c_2c_3c_4c_5 + c_2c_3c_4c_5 + c_3c_4c_5 + c_3c_5 + c_2c_4c_5 + c_2 + c_3c_5 + c_2 + c_2c_3c_4 + c_2$$

and

$$true + true + c_2c_3 + c_2c_3c_4c_5 + c_2c_3 + c_2c_3c_4c_5 + c_2c_3c_4c_5 + c_2c_3c_4c_5 + c_5 + c_2c_3c_4c_5 \leq c_1.$$

Using the rules of boolean algebra, these two precedence relationships can be simplified yielding

$$c_1 \leq c_2 + c_3c_5 \qquad and \qquad true \leq c_1.$$

The second precedence relationship is always satisfied and can be ignored. Similarly, the result of theorem 1, simplifying the logical expressions, and deleting those precedence relationships that have "true" on either side, we obtain four additional precedence relationships. The resulting set of precedence relationships is:

$$c_1 \leq c_2 + c_3c_5 \quad c_2 \leq c_1 + c_3c_4 \quad c_3 \leq c_1c_5 + c_2c_4 \quad c_4 \leq c_5 + c_2c_3 \quad c_5 \leq c_4 + c_1c_3.$$

This set of precedence relationships contains redundances, and it can be shown to be equivalent to:

$$c_3 \leq c_1c_5 + c_2c_4.$$

The precedence relations derived in this manner provide an implicit representation of the ordering constraints imposed by the geometric and mechnical properites of the assembly. A real-time planner can check the feasibility of a next step operation by testing against the precedence relations rather than replanning the entire sequence or checking against an exhaustive representation of plans. This approach provides one example of a plan representation derived for a complex planning problem which yields a representation consistent with an implicit decision-maker for real-time applications.

ACKNOWLEDGEMENTS

Preparation of this paper was supported in part by the NASA Center for Intelligent Systems for Space Exploration and in part by the New York State Center for Advanced Technology in Automation and Robotics at Rensselaer Polytechnic Institute. The work on precedence relations for assembly planning was supported by the Jet Propulsion Laboratory, and results have appeared previously in [13].

REFERENCES

1. Saridis, G. N., "Toward the Realization of Intelligent Controls", *Proc. of the IEEE* 67 August, 1979.
2. Fikes, R., and N. Nilsson, "STRIPS: A New Application of Theorem Proving to Problem Solving," *Artificial Intelligence* 2, pp. 189–208, 1972.
3. Sacerdoti, E.D., "Planning in a Hierarchy of Abstraction Spaces," *Third International Joint Conference on Artificial Intelligence*, pp. 412–422, Stanford Research Institute Publications, 1973.
4. Wilkins, D., "Domain-independent Planning: Representation and Plan Generation," *Artificial Intelligence* 22 PP. 269–301, 1984.
5. Chapman, D., "Planning for Conjunctive Goals," *Artificial Intelligence* 32 pp. 333–377, 1987.
6. Korf, R.E., " Planning as Search: A Quantitative Approach", *Artificial Intelligence* 33 pp. 65–88, 1987.
7. Lozano-Perez, T., and M. Wesley, "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Comm. ACM* 22 pp. 560–570, 1979.

8. Khatib, O., "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal of Robotics and Automation* **RA-3**, 1987.
9. Verbeek, P., L. Dorst, B. Verwer, and F. Groen, "Collision Avoidance and Path Finding through Constrained Distance Transformation in Robot State Space", in *Proc. International Intelligent Autonomous Systems Conference* (L. Hertzberger and F. Groen, eds.), pp. 627–634, Amsterdam, Netherlands, North Holland, December, 1986.
10. Homem de Mello, L. S., and A. C. Sanderson, "AND/OR Graph Representation of Assembly Plans," *Proc. of the Fifth National Conference on Artificial Intelligence AAAI-86* pp. 1113–1119, Morgan Kaufman, 1986.
11. Homem de Mello, L. S., and A. C. Sanderson, "Task Planning and Control Synthesis for Flexible Assembly Systems," in *Machine Intelligence and Knowledge Engineering for Robotic Applications* NATO ASI Series, *Vol. F33*, Ed. A.K.C.Wong and A.Pugh, Springer-Verlag, Berlin, 1987.
12. Homem de Mello, L. S., and A. C. Sanderson, "Automatic Generation of Mechanical Assembly Sequences," Technical Report CMU-RI-TR-88–19, The Robotics Institute, Carnegie Mellon University, December, 1988.
13. Homem de Mello, L. S., and A. C. Sanderson, "Precedence Relationship Representations of Mechanical Assembly Sequences," in *Proc. 2nd NASA Workshop on Space Telerobotics* Pasadena, CA, January, 1989.
14. Homem de Mello, L. S., and A. C. Sanderson, "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences," *IEEE Transactions on Robotics and Automation*, in press, 1989.
15. Homem de Mello, L. S., *Task Sequence Planning for Robotic Assembly*, PhD Dissertation, Electrical and Computer Engineering Department, Carnegie Mellon University, May, 1989.
16. Bourjault, A., *Contribution a une Approche Methodologique de L'Assemblage Automatise: Elaboration Automatique des Sequences Operatoires*, These d'Etat, Universite de Franche-Comte, Besancon, France, November, 1984.
17. DeFazio, T. L., and D. E. Whitney, "Simplified Generation of All Mechanical Assembly Sequences", *IEEE Journal of Robotics and Automation* RA-3(6), pp. 640–658, December, 1987. See Corrections, same journal RA-4(6), pp. 705–708, December, 1988.